

WANNENMACHER  
Lucas  
Candidat n° : 7022

# Contrôle de la propagation des ondes sonores basses fréquences



# Sommaire

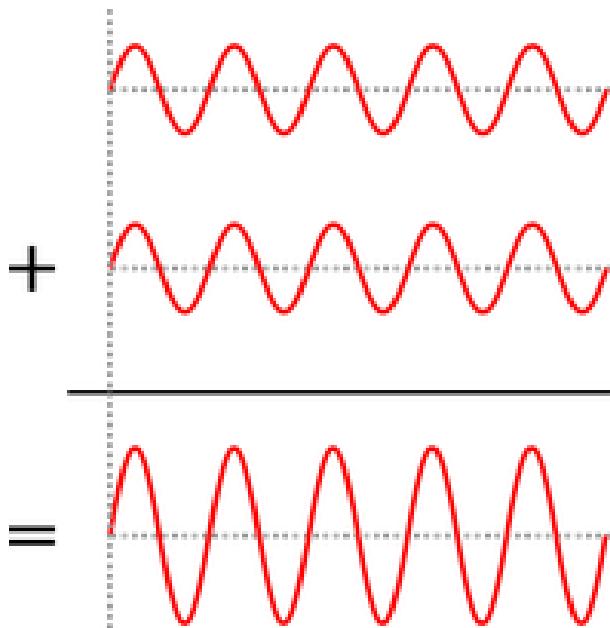
Comment optimiser le contrôle de la directivité des ondes sonores basses fréquences?

- I. Introduction et position du problème
- II. Résolution analytique
- III. Modélisation spatiale
- IV. Confrontation avec la réalité

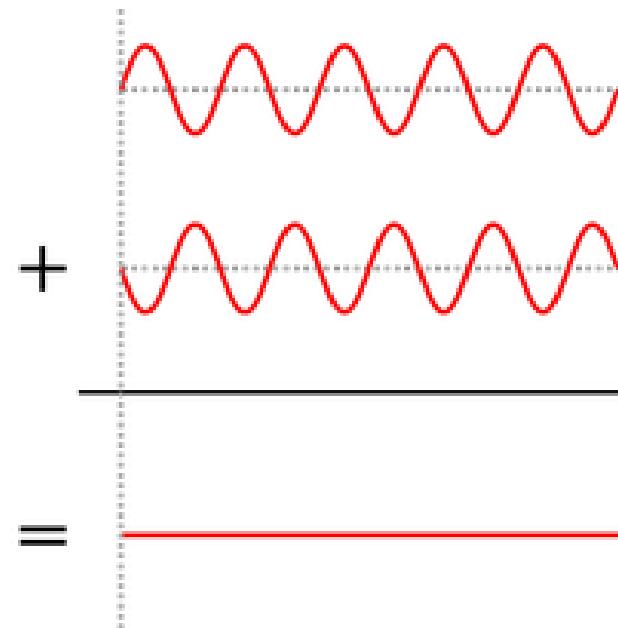
## I. Introduction et position du problème

### I.a. Phénomènes d'interférences

Interférences  
constructives

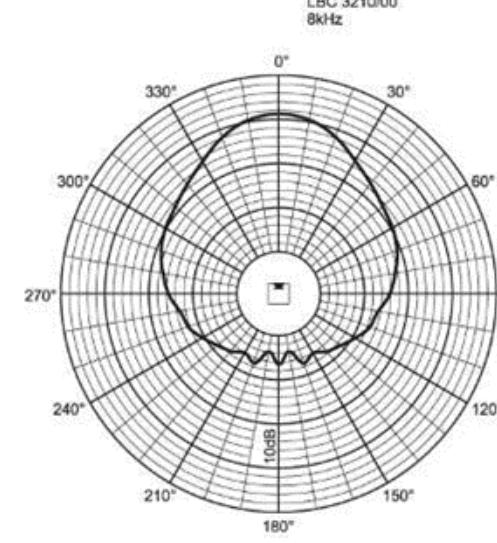
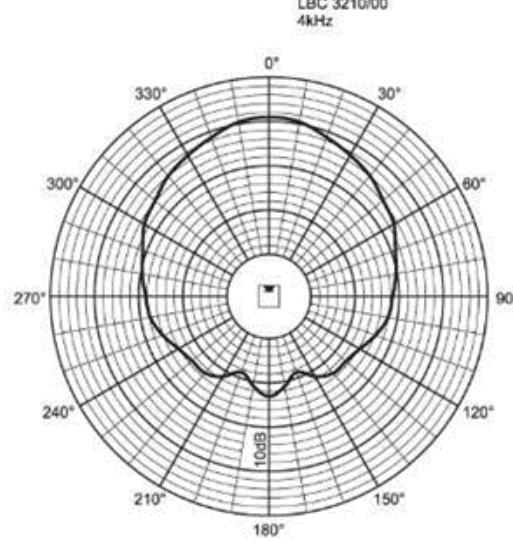
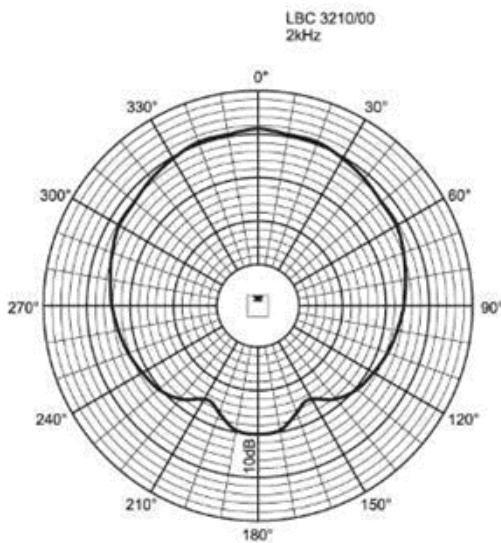
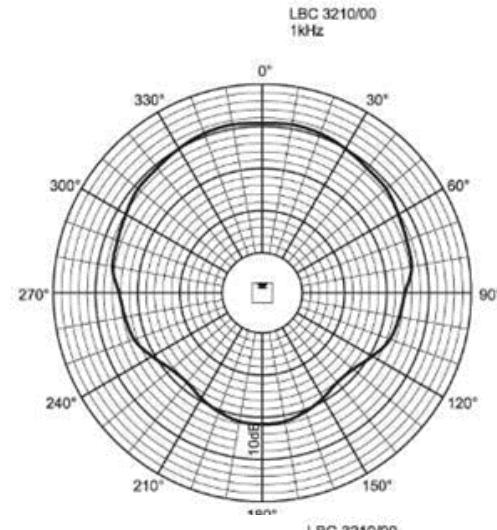
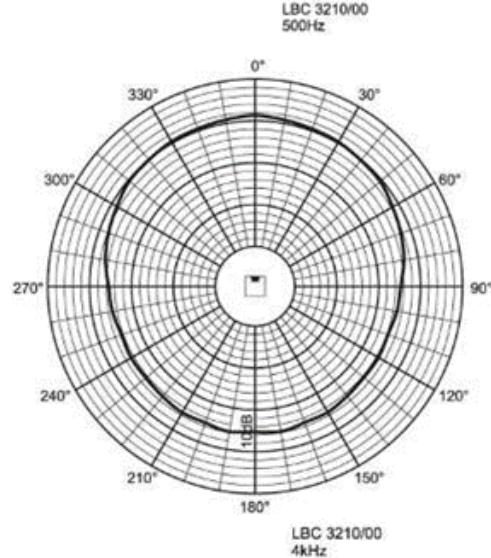
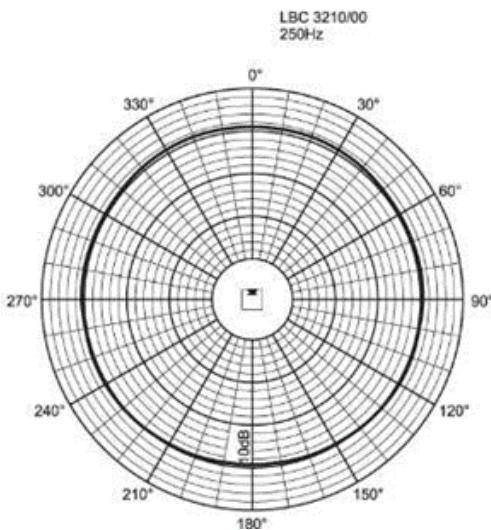


Interférences  
destructives

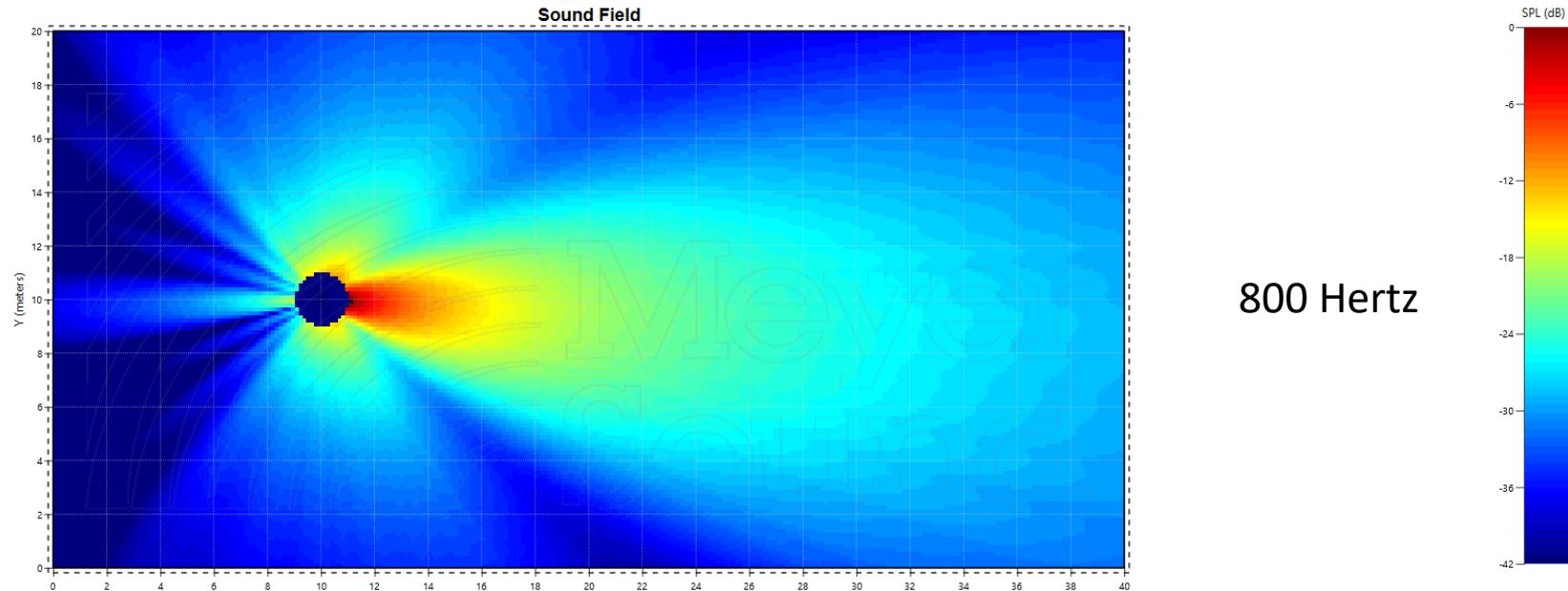
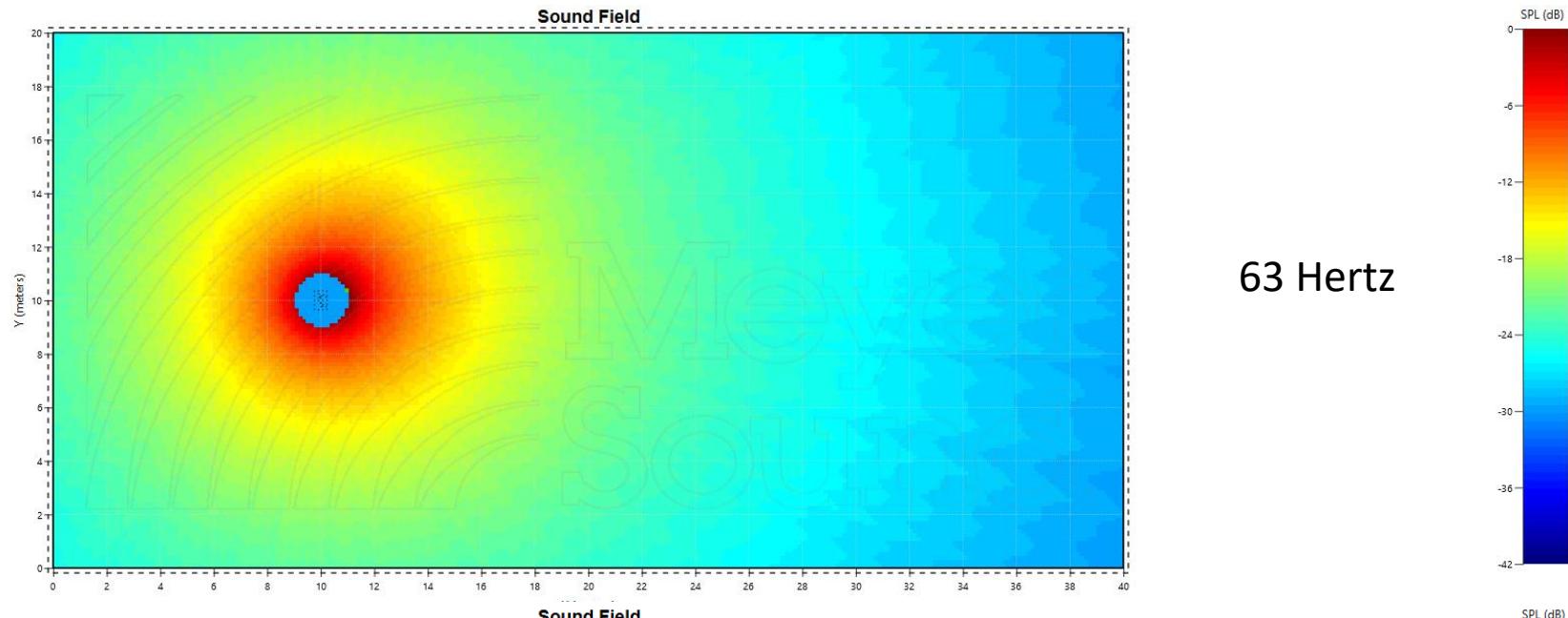


\*Source disponible en annexe

## I.b. Limitation aux basses fréquences



\*Source disponible en annexe

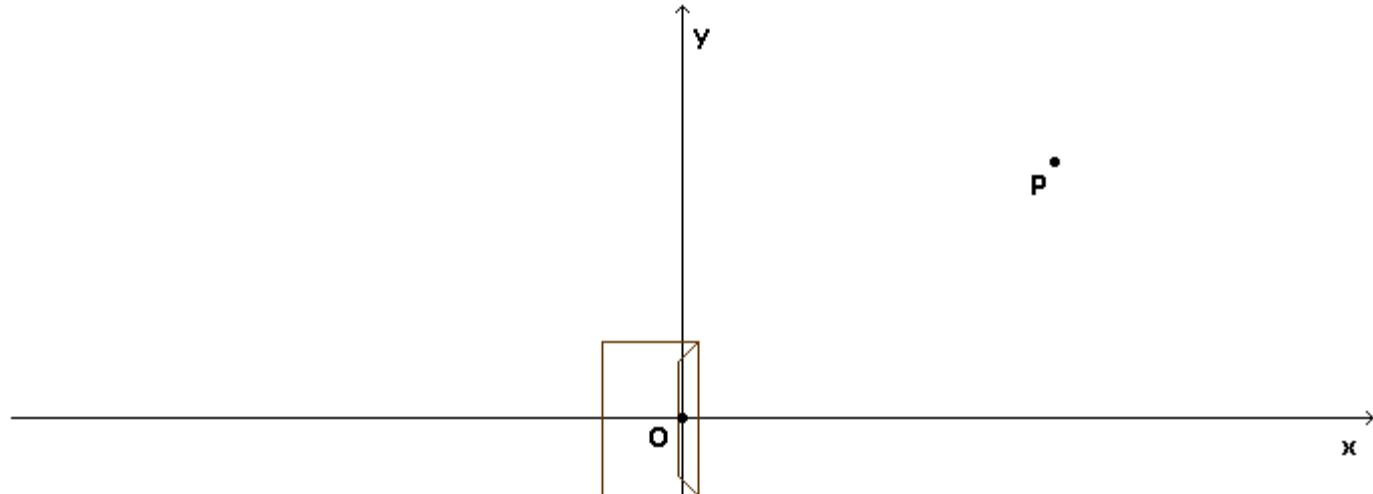


## II. Résolution analytique

## II.1 : Un seul haut parleur centré

Expression du niveau sonore au point P en fonction des coordonnées de P( $x_P, y_P$ ):

$$L(P) = L_{1m} - 20 * \log(\sqrt{x_P^2 + y_P^2})$$



Hypothèses et notations :

- Source ponctuelle et omnidirectionnelle
- $L_{1m}$  représente le niveau sonore à 1m de la source sonore

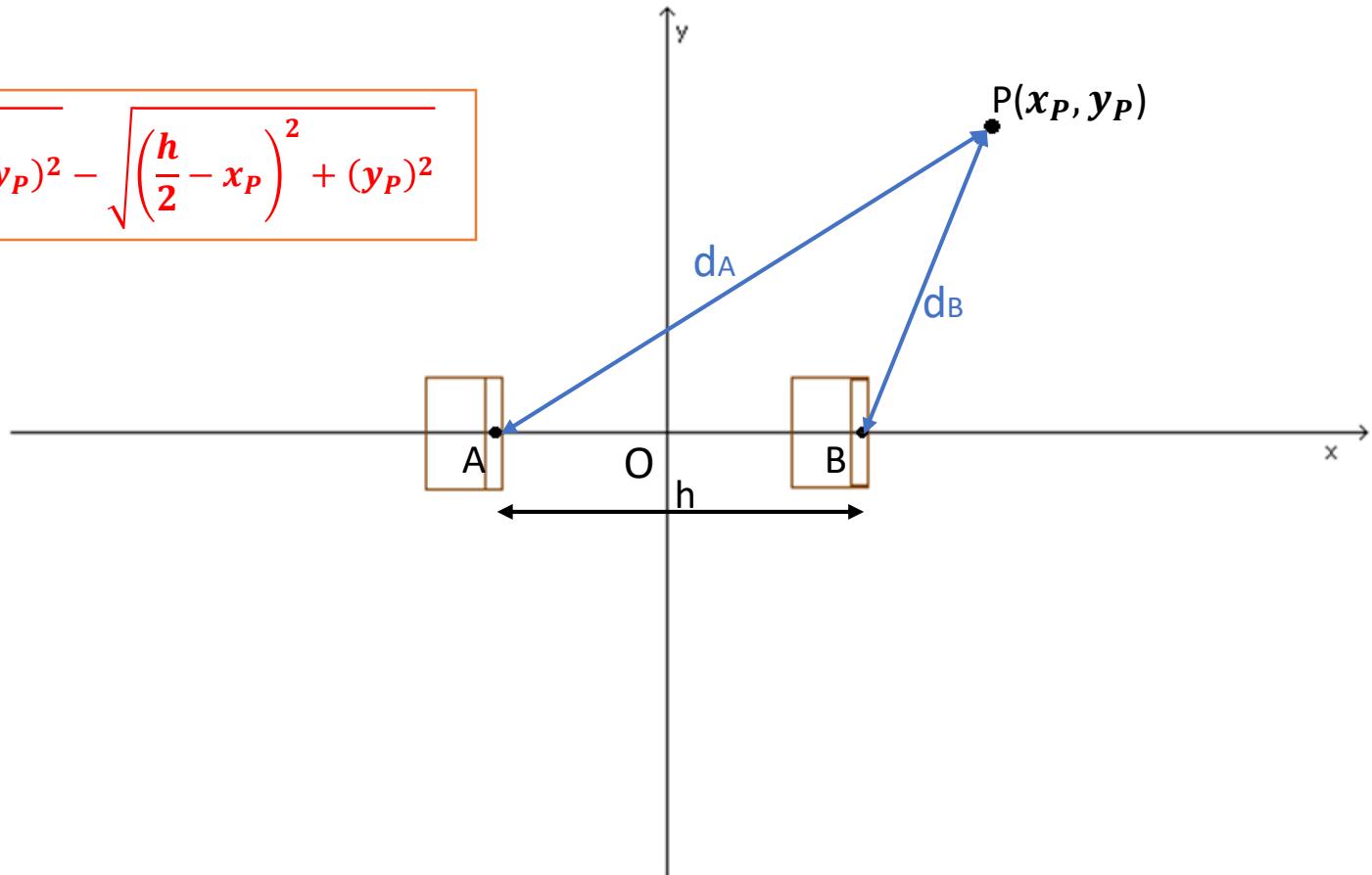
## II.2: Généralisation à deux haut-parleurs déphasés

Au point  $P(x_p, y_p)$  :

Différence de marche  $\Delta l$  :

$$\Delta l = d_A - d_B$$

$$\Delta l = \sqrt{\left(-\frac{h}{2} - x_p\right)^2 + (y_p)^2} - \sqrt{\left(\frac{h}{2} - x_p\right)^2 + (y_p)^2}$$



\*Détails des calculs disponibles en annexe

Déphasage dû à la différence de marche :

$$\varphi = \frac{\Delta l * f * 2\pi}{V_{son}}$$

Niveaux sonores au point P :

- en isolant le haut-parleur B :  $L_A(P) = L_{A1m} - 20 \log (\sqrt{(-\frac{h}{2} - x_P)^2 + (y_P)^2})$
- en isolant le haut-parleur A :  $L_B(P) = L_{B1m} - 20 \log (\sqrt{(\frac{h}{2} - x_P)^2 + (y_P)^2})$

Intensités acoustiques au point P :

- en isolant le haut-parleur B :  $I_A(P) = 10^{L_A(P)} * I_0$
- en isolant le haut-parleur A :  $I_B(P) = 10^{L_B(P)} * I_0$

Formule de superpositions de signaux périodiques déphasés \*:

$$u(t) + v(t) = \sqrt{(U + V \cos(\varphi))^2 + (V \sin(\varphi))^2} * \cos(\omega t)$$

avec  $u(t) = U \cos(\omega t)$  et  $v(t) = V \cos(\omega t + \varphi)$

\*Source disponible en annexe

Intensité sonore acoustique résultante au point P :

$$I_R(P) = \sqrt{(I_A + I_B \cos(\varphi))^2 + (I_B \sin(\varphi))^2}$$

\*Détails des calculs disponibles en annexe

Niveau sonore résultant au point P :

$$L_r(P) = 10 * \log \left( \frac{I_R(P)}{I_0} \right)$$

### Ajout d'un délai $\Delta t$ à un haut-parleur :

- La différence de marche est virtuellement modifiée :

$$\Delta l = (l_2 - l_1) + V_{son} * \Delta t$$

Les expressions de  $\varphi$ ,  $I_R(P)$  et  $L_R(P)$  sont donc successivement modifiées.

On peut donc exprimer  $L_r(P)$  résultant en fonction de  $h$ ,  $x_P$ ,  $y_P$  et  $\Delta t$  \*

\*Détails des calculs disponibles en annexe

## II. Modélisation spatiale

II.a. Algorithme Python

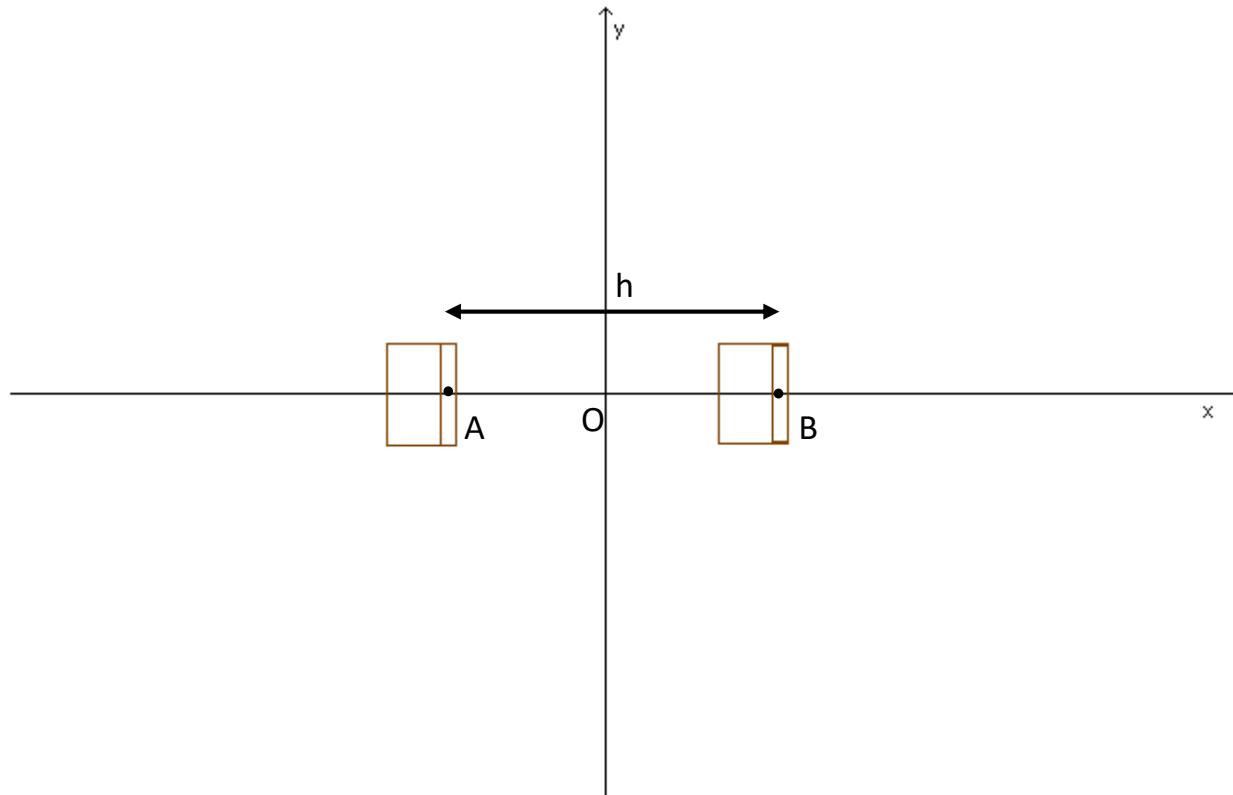
II.b. Exploitation et recherche de montages  
optimaux

II.c. Représentation sous le logiciel MAPP XP

## II.a. Algorithme Python

```
# C:\Users\lucas\Desktop\TIPE\programme final.py
001| ##Bibliothèque
002|
003| import matplotlib.pyplot as plt
004| import numpy as np
005|
006| ##Fonctions
007| #fonctions servant au calcul du niveau sonore
008|
009| def niveau_sonore_1HP(X,Y,HP): #HP contient les caractéristiques d'un haut-
parleur : HP = [abscisse,ordonnée,L1m, délai]
010|     d = np.sqrt((X-HP[0])**2+(Y-HP[1])**2)
011|     L = HP[2]-20*np.log10(d)
012|     return L
013|
014| def difference_de_marche(X,Y,HP1,HP2):
015|     d1 = np.sqrt((HP1[1]-Y)**2+(HP1[0]-X)**2)
016|     d2 = np.sqrt((HP2[1]-Y)**2+(HP2[0]-X)**2)
017|     différence = d2-d1
018|     return différence
019|
020| def distance_delai(HP1,HP2):
021|     delta_t = HP2[3]-HP1[3]
022|     d = Vson*delta_t
023|
024|     return d
025|
026| def dephasage(f,X,Y,HP1,HP2):
027|     Vson = 340
028|     d = difference_de_marche(X,Y,HP1,HP2)+distance_delai(HP1,HP2)
029|     phi = (d*f**2*np.pi)/Vson
030|     phi = phi%(2*np.pi)
031|     return phi
032|
033| def niveau_sonore_2HP(X,Y,HP1,HP2): #niveau soore généralisé à 2HP
034|     L1 = niveau_sonore_1HP(X,Y,HP1)
035|     L2 = niveau_sonore_1HP(X,Y,HP2)
036|     phi = dephasage(f,X,Y,HP1,HP2)
037|     I1 = 10**(L1/10)*(10**(-12))
038|     I2 = 10**(L2/10)*(10**(-12))
039|     I_resultant = np.sqrt((I1+I2*np.cos(phi))**2+(I2*np.sin(phi))**2)
040|     L_resultant = 10*np.log10(I_resultant/(10**(-12)))
041|     return L_resultant
042|
066| #fonctions servant à la représentation graphique
067|
068| def cartographie(AmplitudeX,AmplitudeY, pas):
069|     Liste_coordonnées = []
070|     X0 = -AmplitudeX/2
071|     Y0 = -AmplitudeY/2
072|     nX = int(AmplitudeX/pas)
073|     nY = int(AmplitudeY/pas)
074|     x = X0
075|     for i in range(nX):
076|         y = Y0
077|         for j in range(nY):
078|             Point = []
079|             Point.append(x)
080|             Point.append(y)
081|             Liste_coordonnées.append(Point)
082|             Point.append(niveau_sonore_2HP(Point[0],Point[1],HP1,HP2)) #modif
083|             y = y + pas
084|         x = x + pas
085|     return Liste_coordonnées
086|
087|
088|     return Liste_L
089|
090| def selection(Liste_L, min, max): #modif
091|     Pts_selectionnés = []
092|     for i in range(len(Liste_L)):
093|         if min<Liste_L[i][2]<max:
094|             Pts_selectionnés.append(Liste_L[i])
095|     return Pts_selectionnés
096|
097|
098|
099| ##Programme principal
100| #Variables
101| f = 63
102| Vson = 340
103| HP1 = [0,-1.35/2,100,0]
104| HP2 = [0,1.35/2,100,0.004]#modif
105|
106| #Tracé
107| L = cartographie(17,28,0.1)
108| S = selection(L,83.9,84.1)
109|
110| abscisses = []
111| ordonnées = []
112|
113| for i in range(len(S)):
114|     abscisses.append(S[i][0])
115|     ordonnées.append(S[i][1])
116|
117|
118| plt.scatter(abscisses,ordonnées, label = "87 dB")
119| plt.scatter(HP1[0],HP1[1], label = "HP1")
120| plt.scatter(HP2[0],HP2[1],label ="HP2")
121| plt.legend()
122| plt.title("cartographie")
123| axis = ("equal")
124| plt.show()
```

## II.b. Exploitation et recherche de montages optimaux



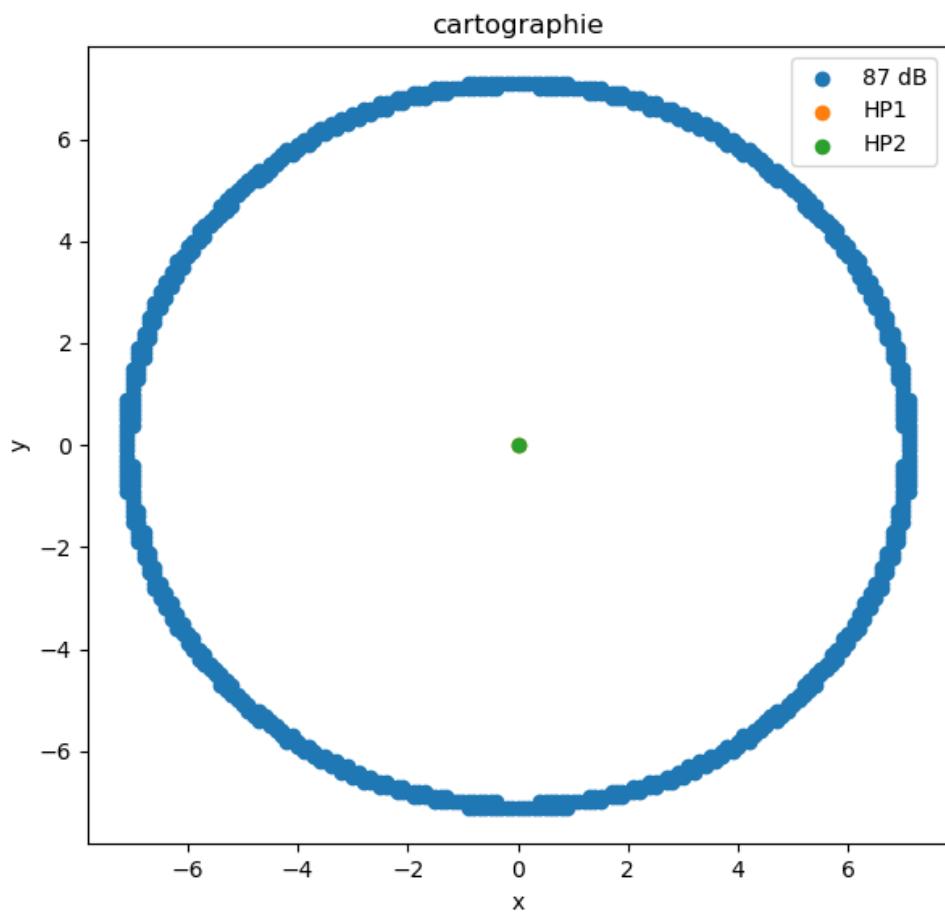
Fréquence : 63 Hz  
Puissance égales :  $L_{1m}$  : 100 dB

Distance  $h$  entre les deux haut-parleurs :

$$h = 0 \text{ m}$$

Délai  $t$  entre les deux haut-parleurs :

$$t = 0 \text{ s}$$

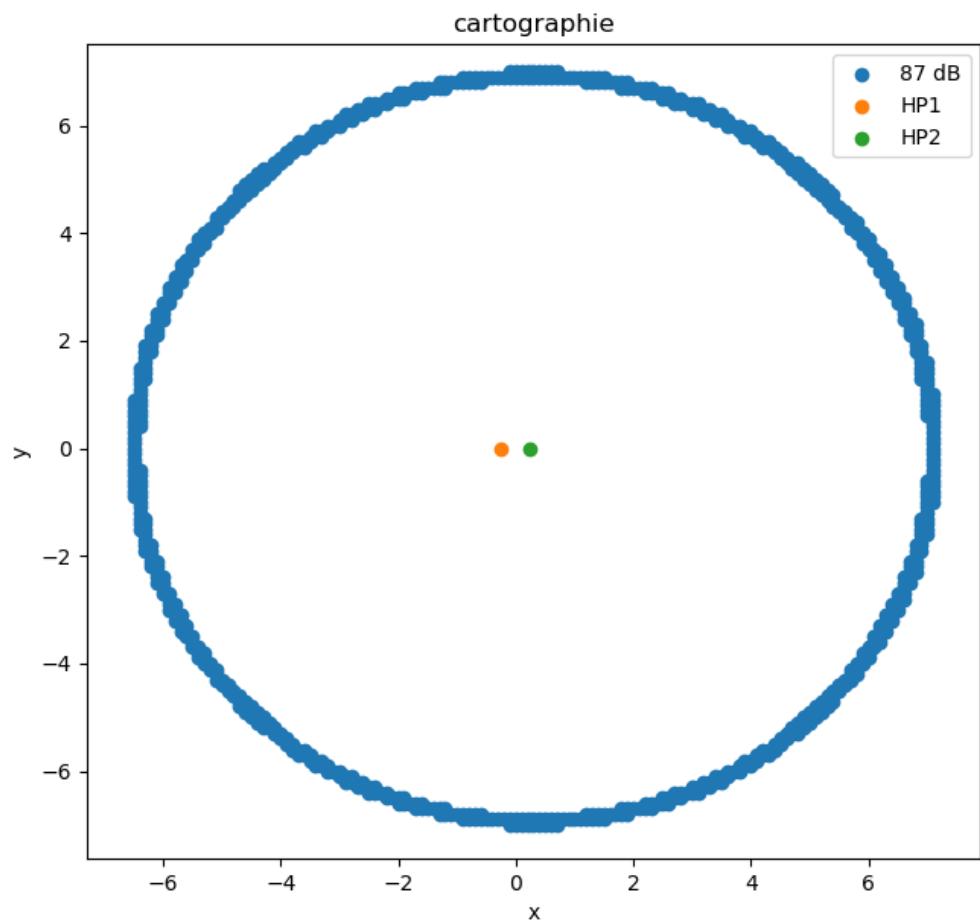


Distance  $h$  entre les deux haut-parleurs :

$$h = 0,5 \text{ m}$$

Délai  $t$  entre les deux haut-parleurs :

$$t = 1,5 \text{ ms}$$

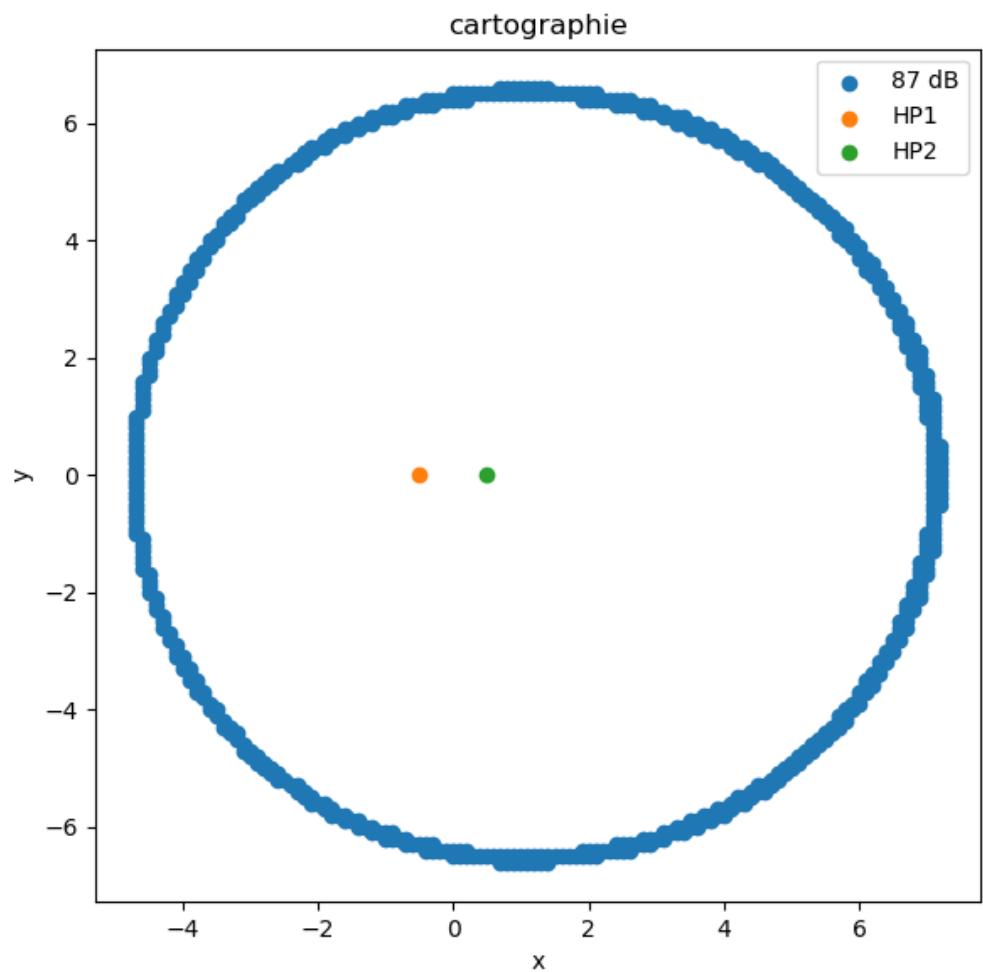


Distance  $h$  entre les deux haut-parleurs :

$$h = 1 \text{ m}$$

Délai  $t$  entre les deux haut-parleurs :

$$t = 2,9 \text{ ms}$$

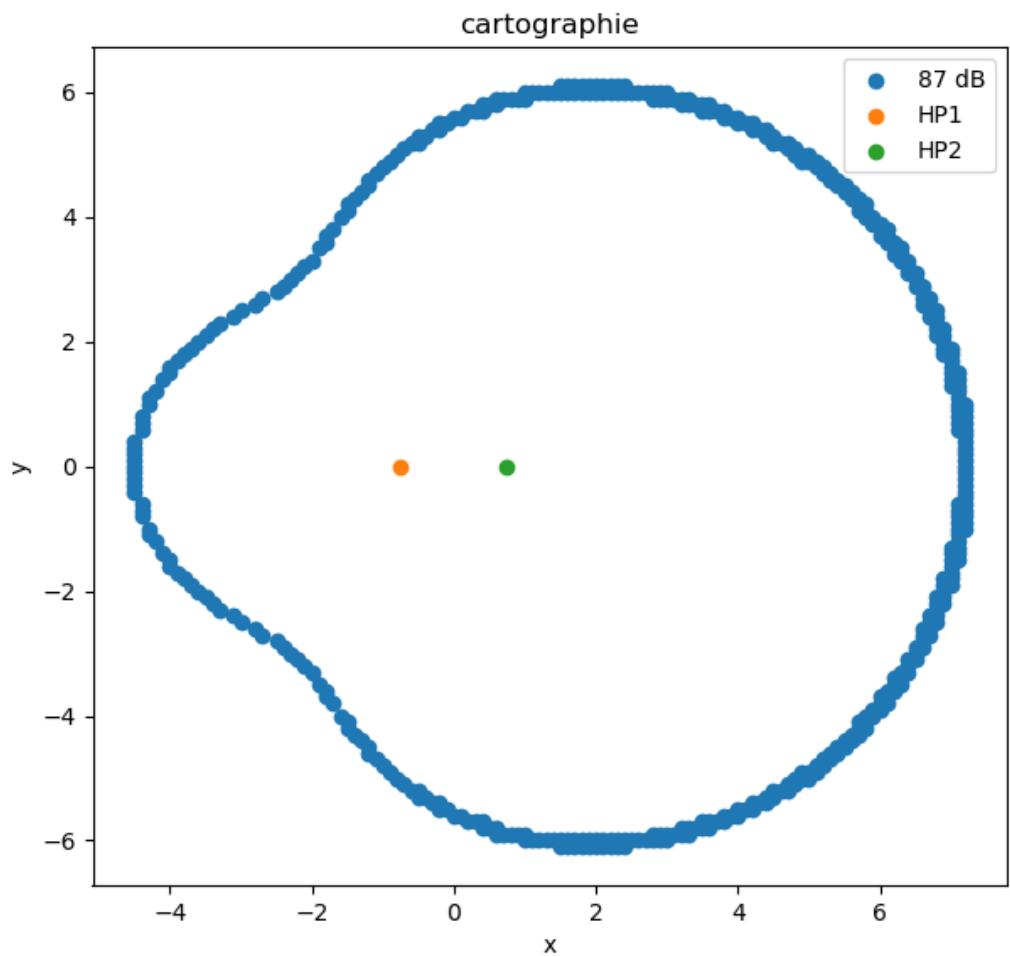


Distance  $h$  entre les deux haut-parleurs :

$$h = 1,5 \text{ m}$$

Délai  $t$  entre les deux haut-parleurs :

$$t = 4,4 \text{ ms}$$

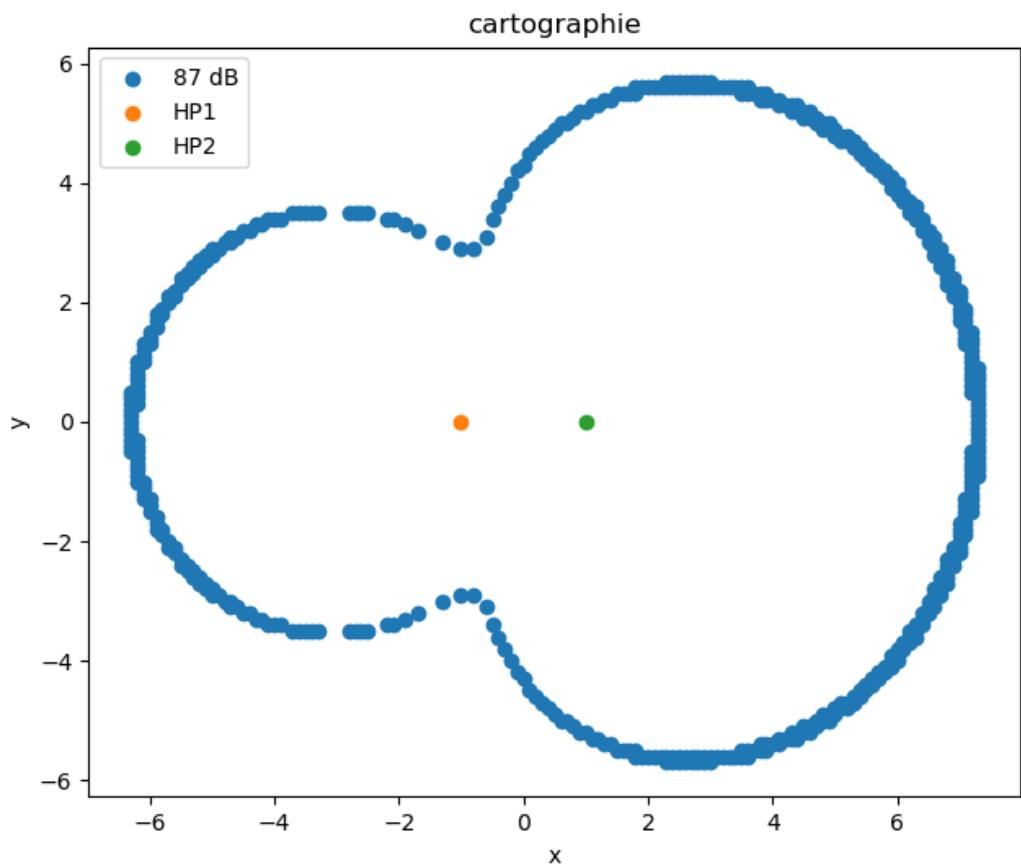


Distance  $h$  entre les deux haut-parleurs :

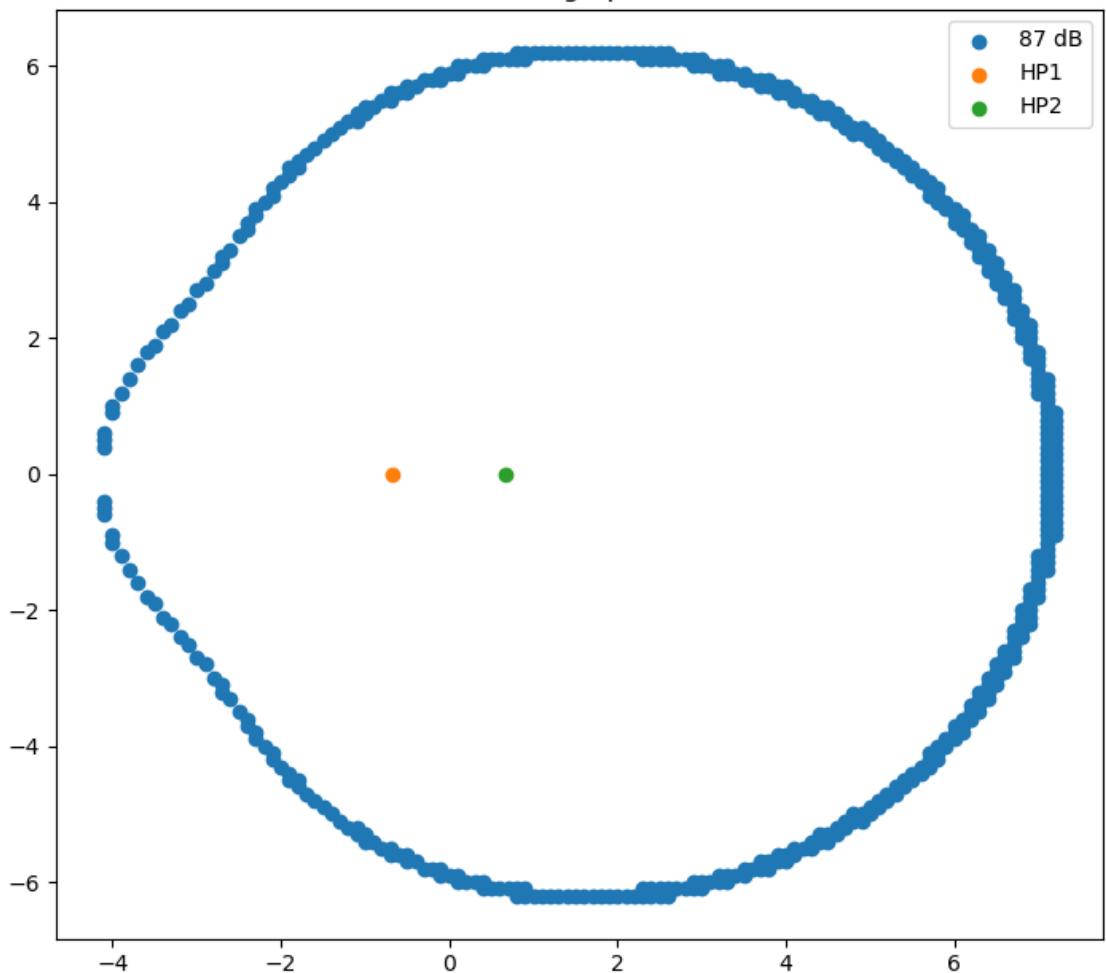
$$h = 2,0 \text{ m}$$

Délai  $t$  entre les deux haut-parleurs :

$$t = 5,9 \text{ ms}$$



cartographie



Distance  $h$  entre les deux haut-parleurs :

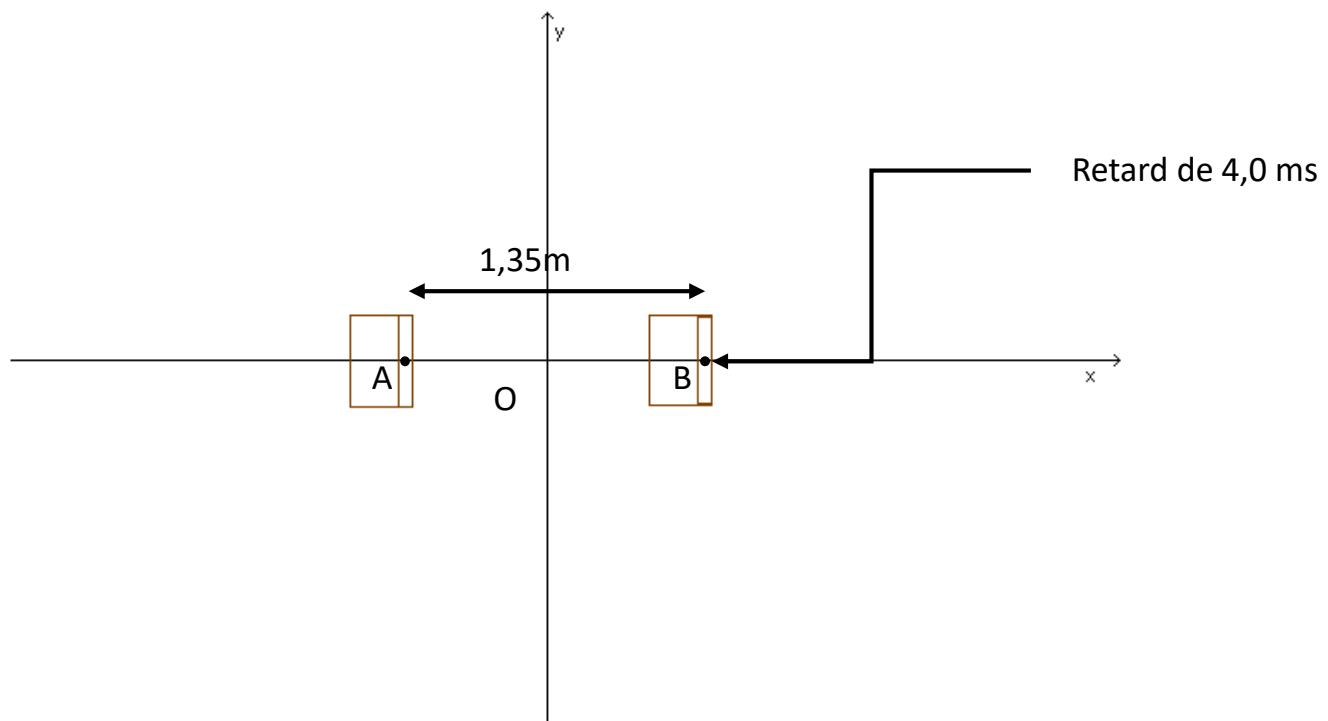
$$h = 1,35 \text{ m}$$

Délai  $t$  entre les deux haut-parleurs :

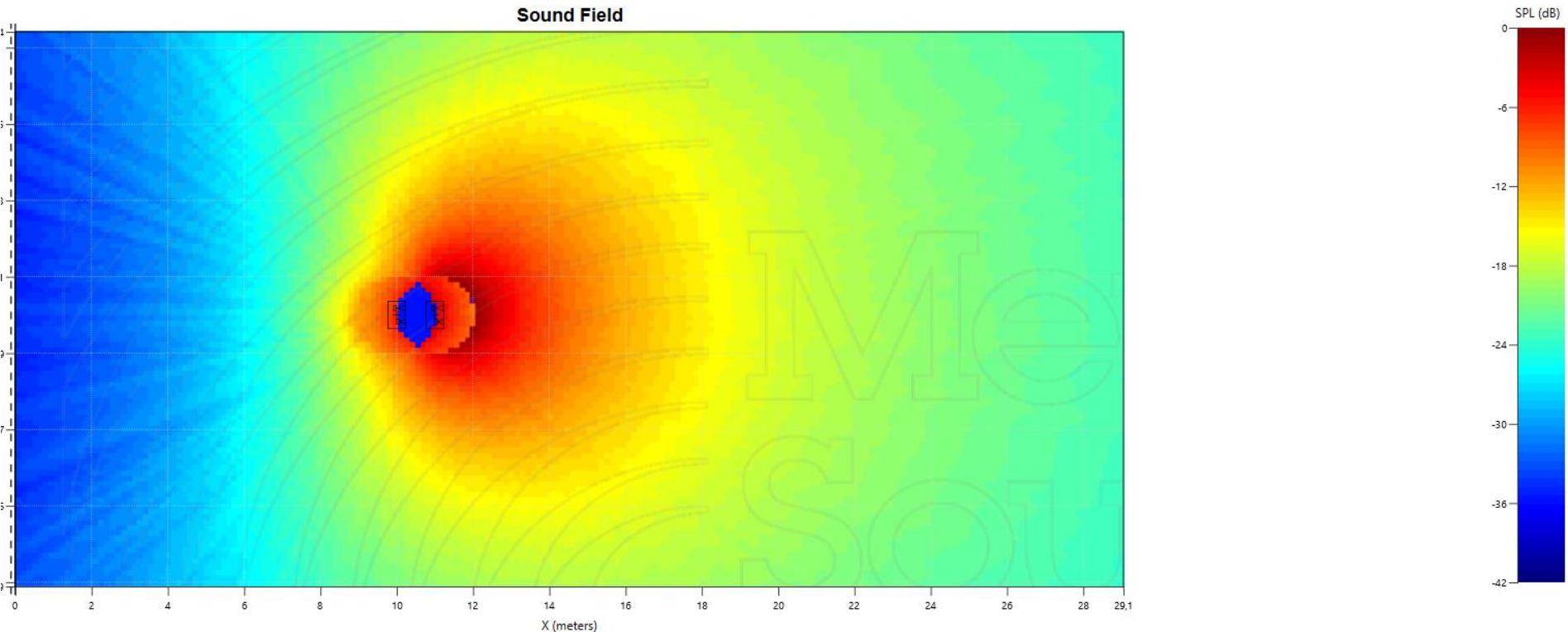
$$t = 4,0 \text{ ms}$$

Configuration retenue :

$$h = 1,35\text{m} = \lambda /4$$
$$t = 4,0 \text{ millisecondes} \Leftrightarrow \text{déphasage de } \pi/2$$

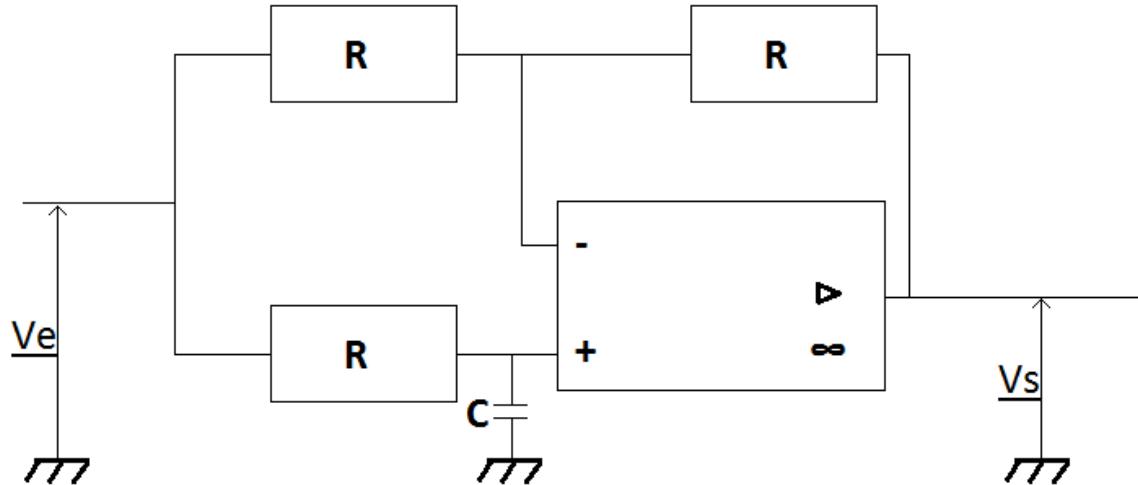


## II.c. Représentation sous le logiciel MAPP XT



### III. Confrontation avec la réalité

#### III.a. Réalisation d'un montage déphaseur



Fréquence : 63 Hz

$$\omega = 396 \text{ rad/s}$$

Souhaitant un déphasage de  $\frac{\pi}{2}$ :

Fonction de transfert :

$$H(j\omega) = \frac{1 - jRC\omega}{1 + jRC\omega}$$

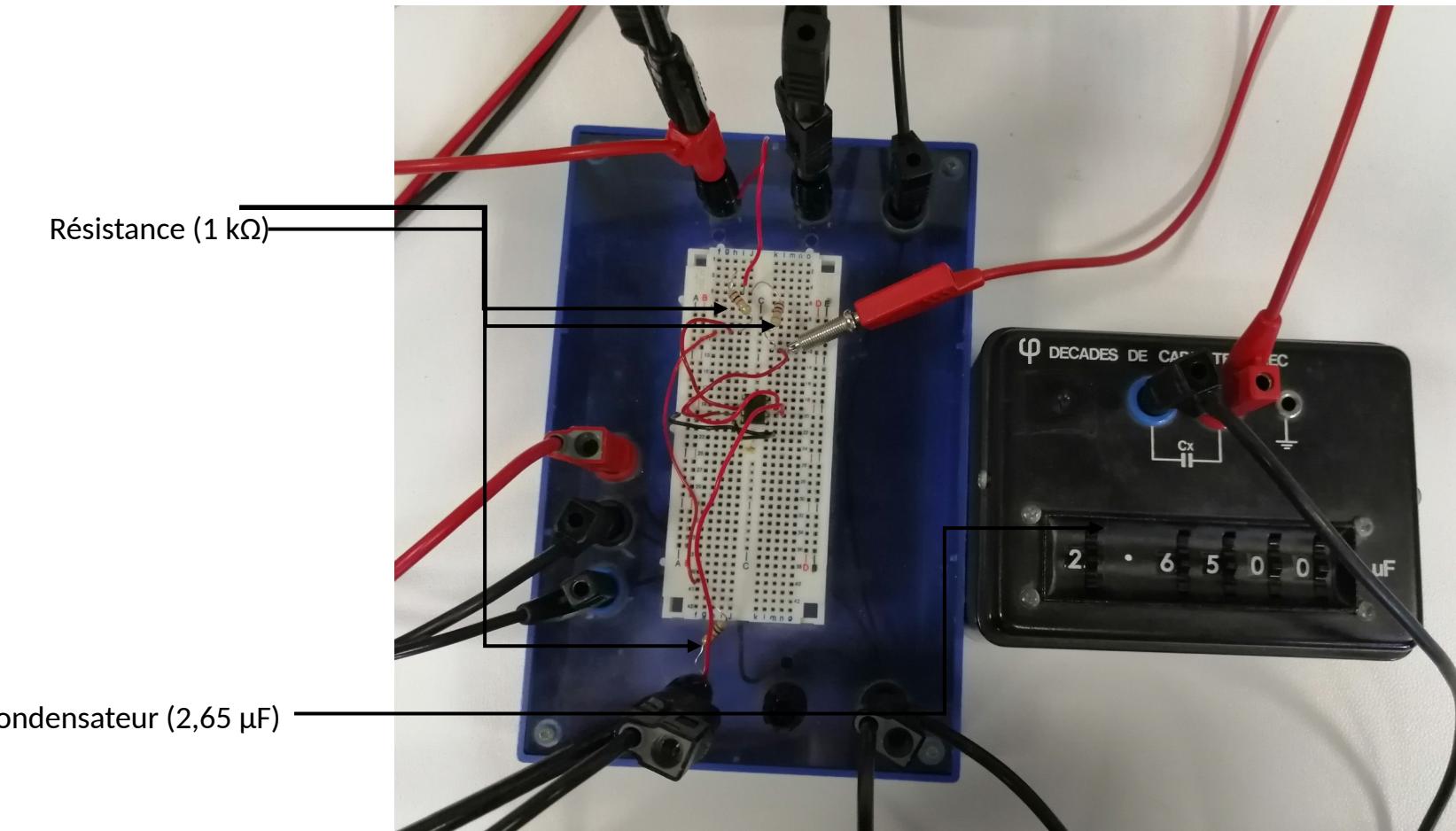
$$R = 1 \text{ k}\Omega$$
$$C = 2,65 \mu\text{F}$$

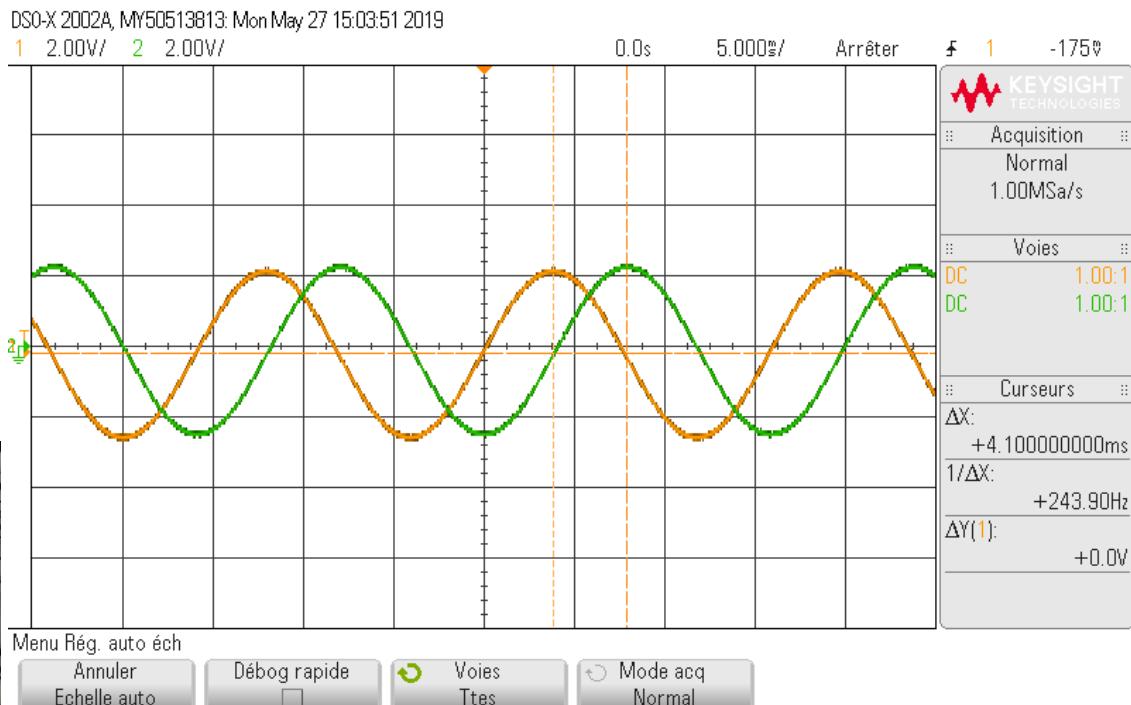
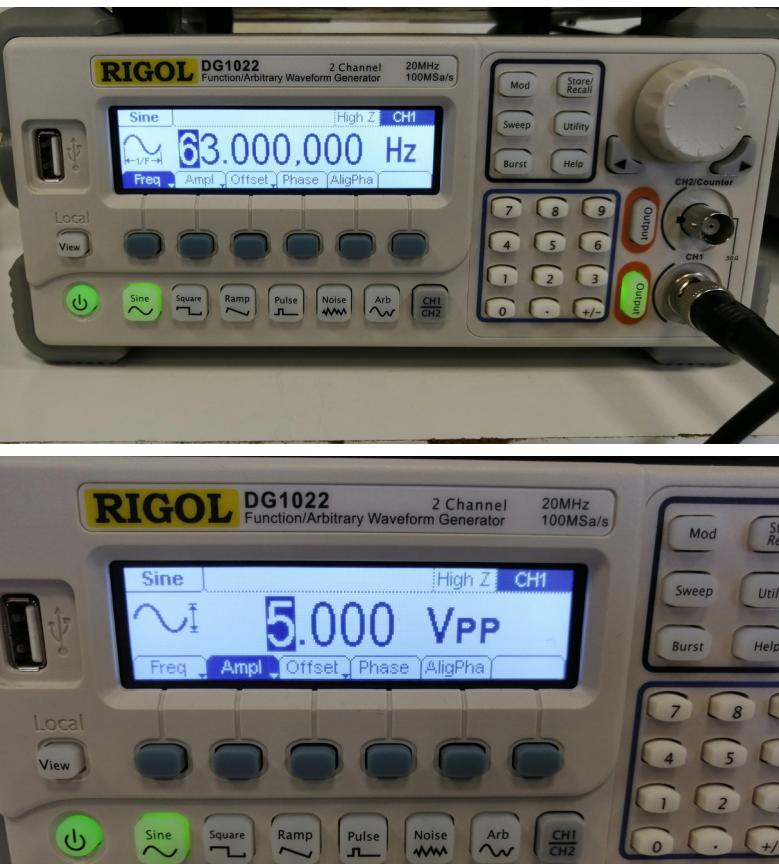
Gain :

1

Déphasage :

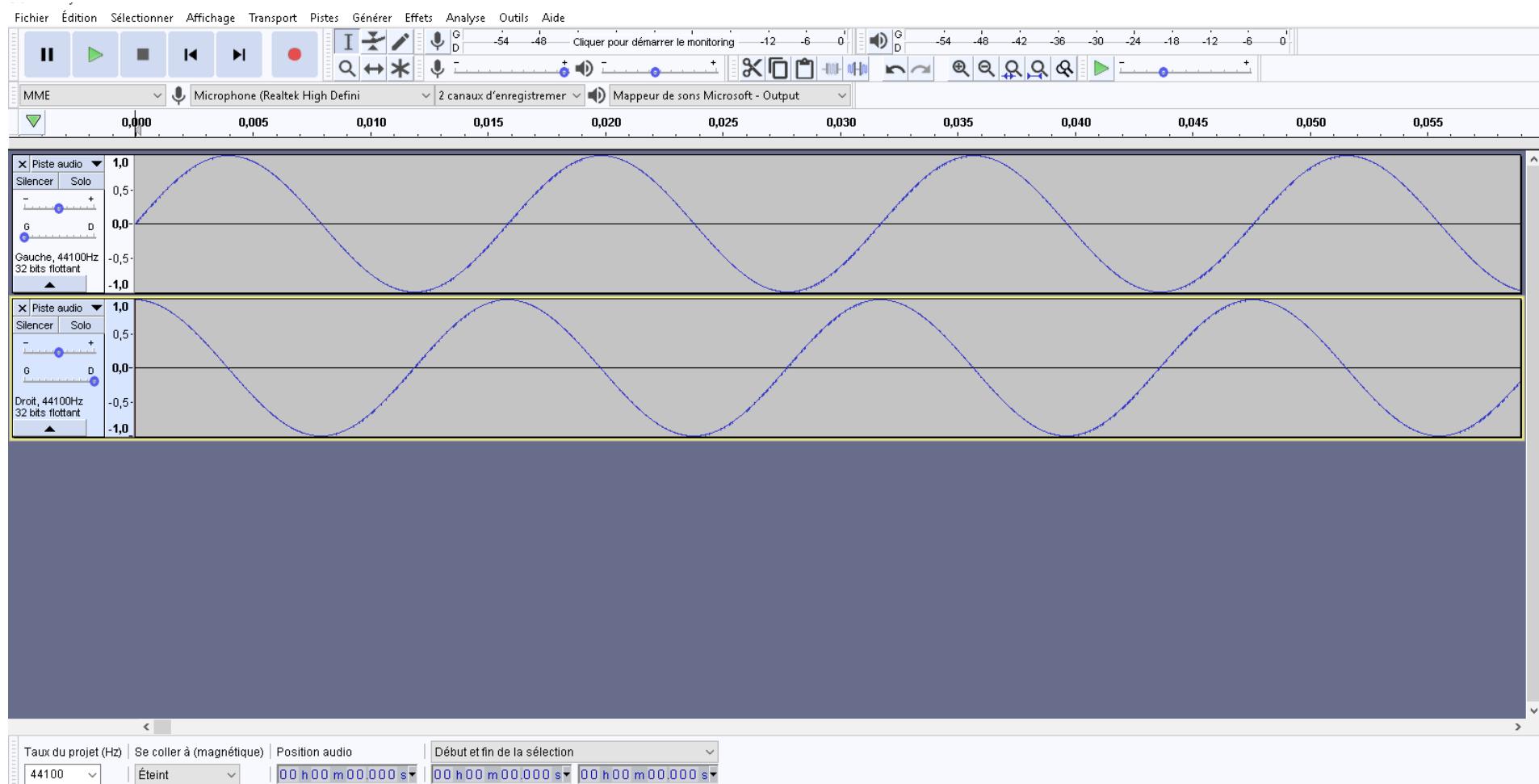
$$2 * \arctan(Rc\omega)$$



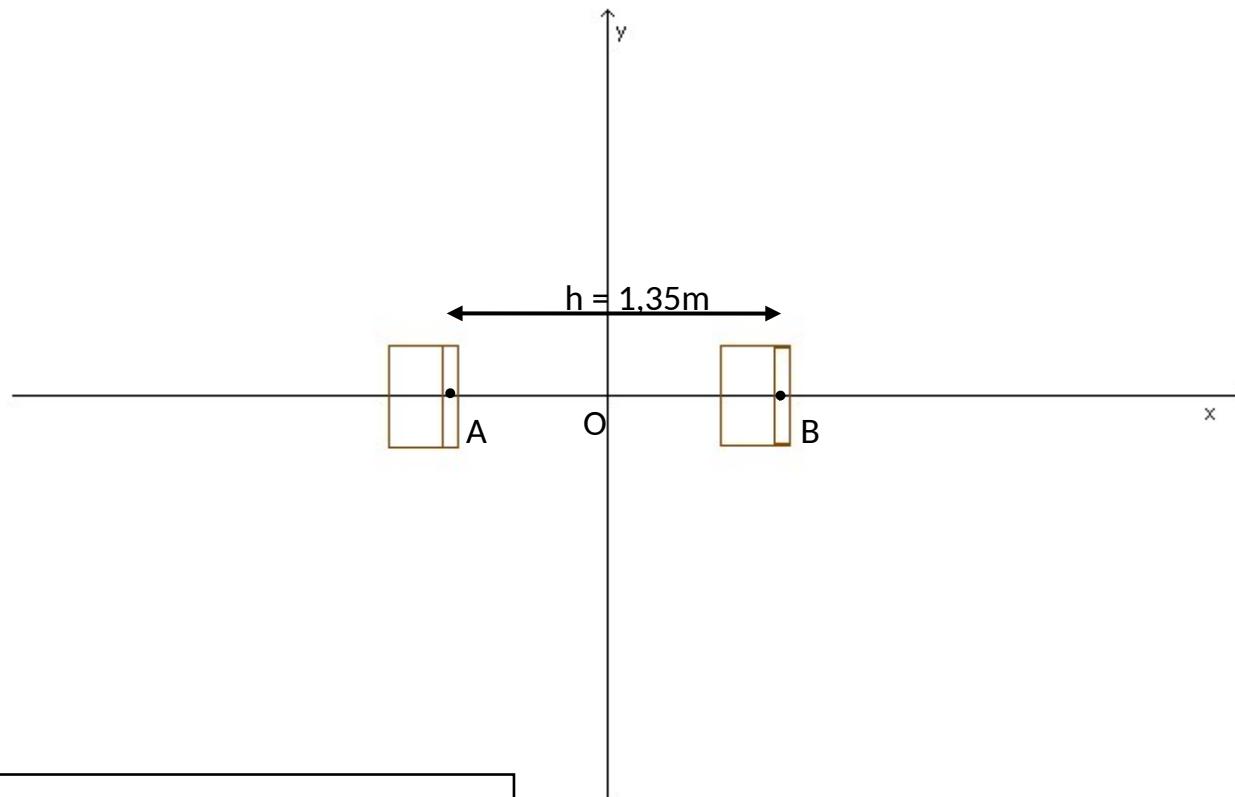


Voie 1 : Entrée  
Voie 2 : Sortie

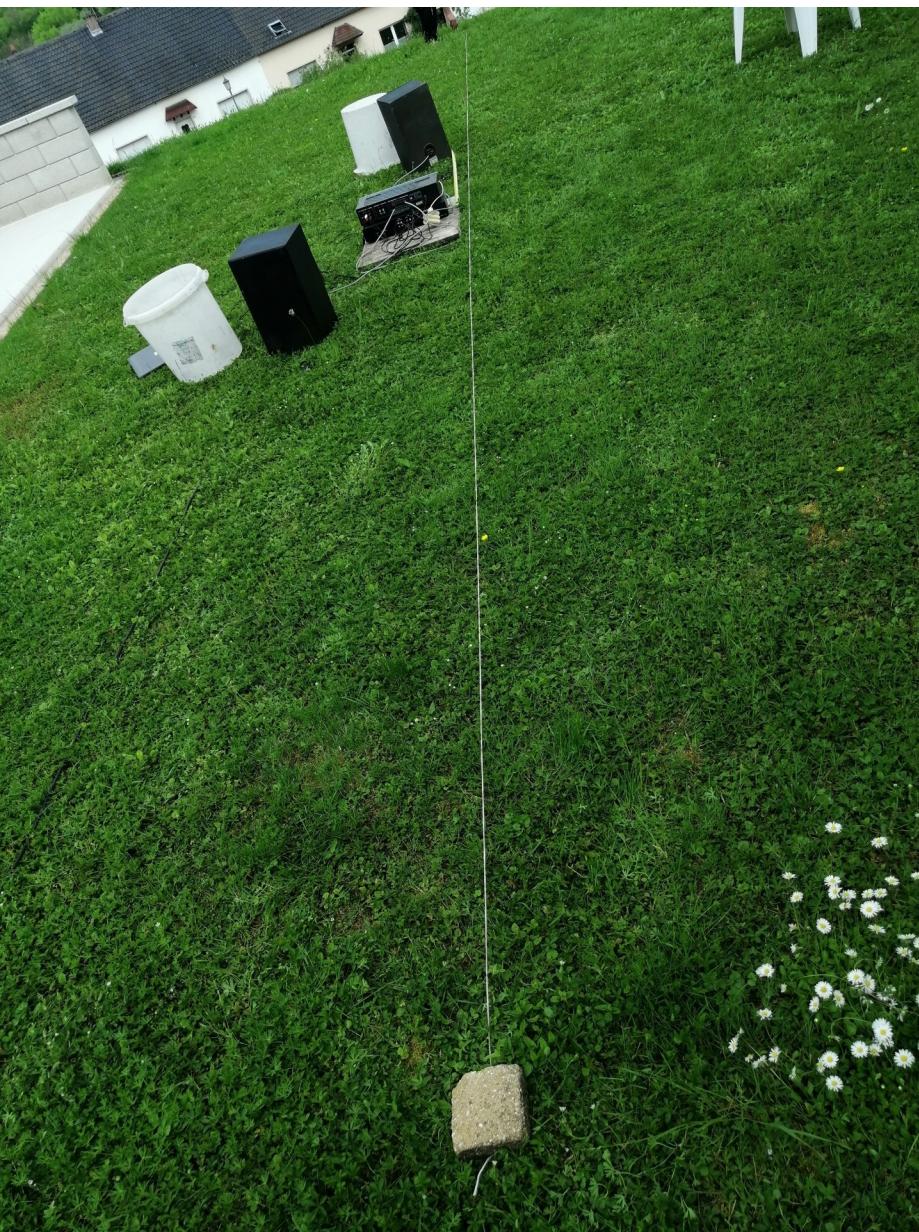
# Alternative : déphasage numérique



## III.b. Mesures expérimentales



63 Hz  
Haut-parleur B retardé de 4,0 ms  
Puissance égales



1,35 m

# Mesures

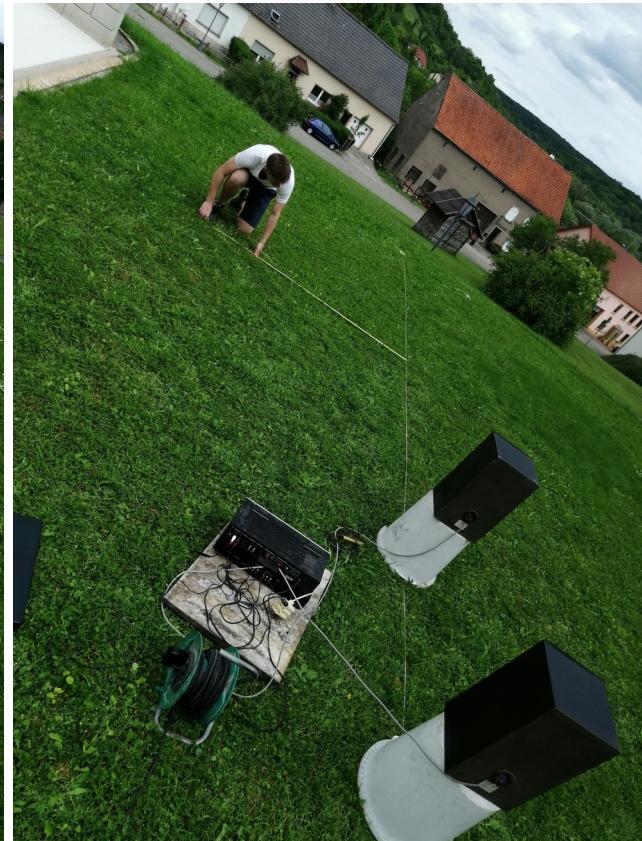
Mesure du niveau sonore  
Relatif à la fréquence 63 Hz



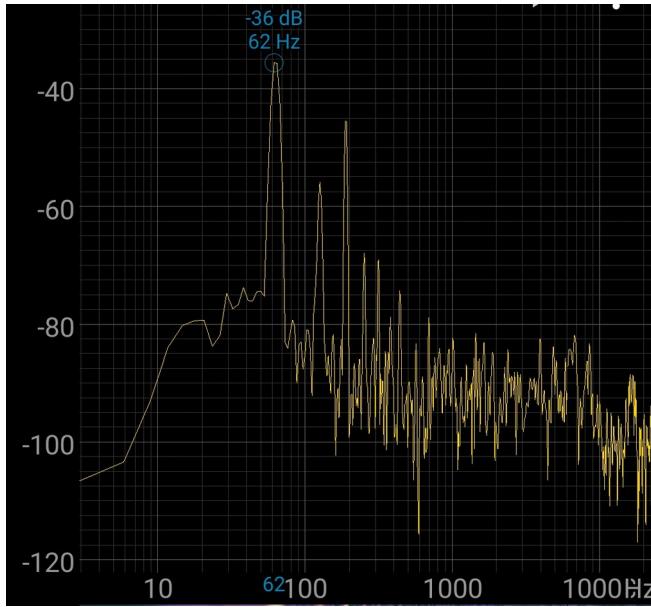
Mesure de l'abscisse



Mesure de l'ordonnée

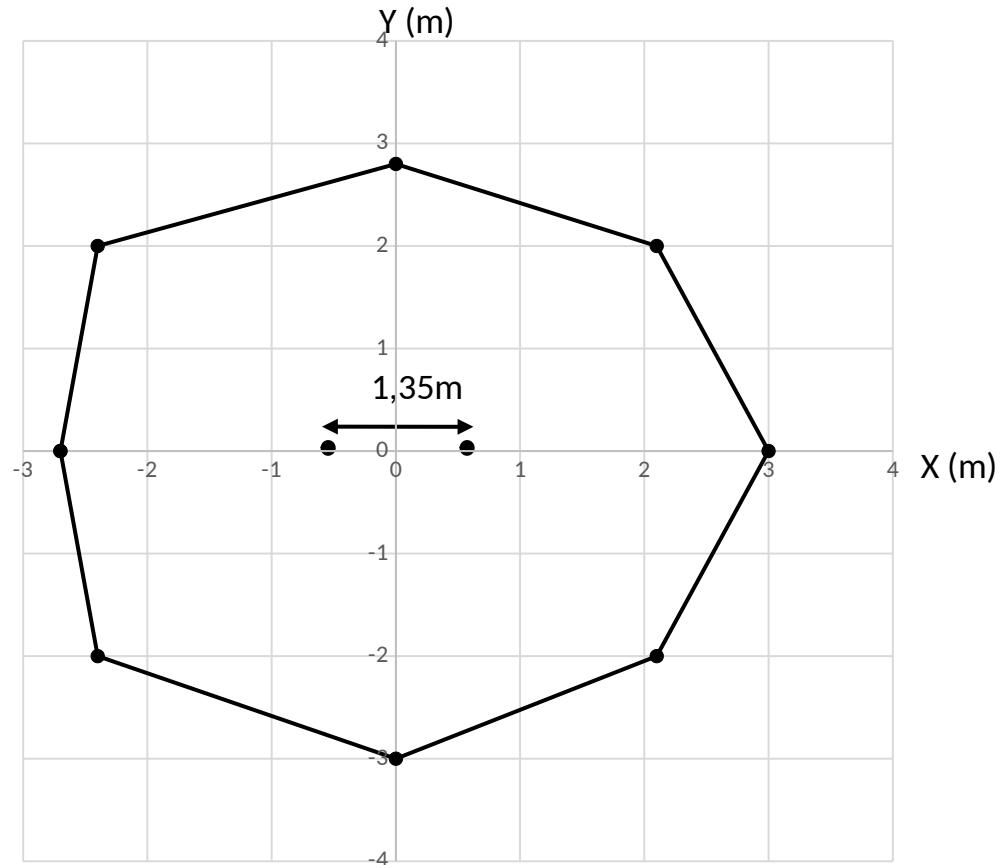


# Signal non déphasé : courbe à niveau sonore constant

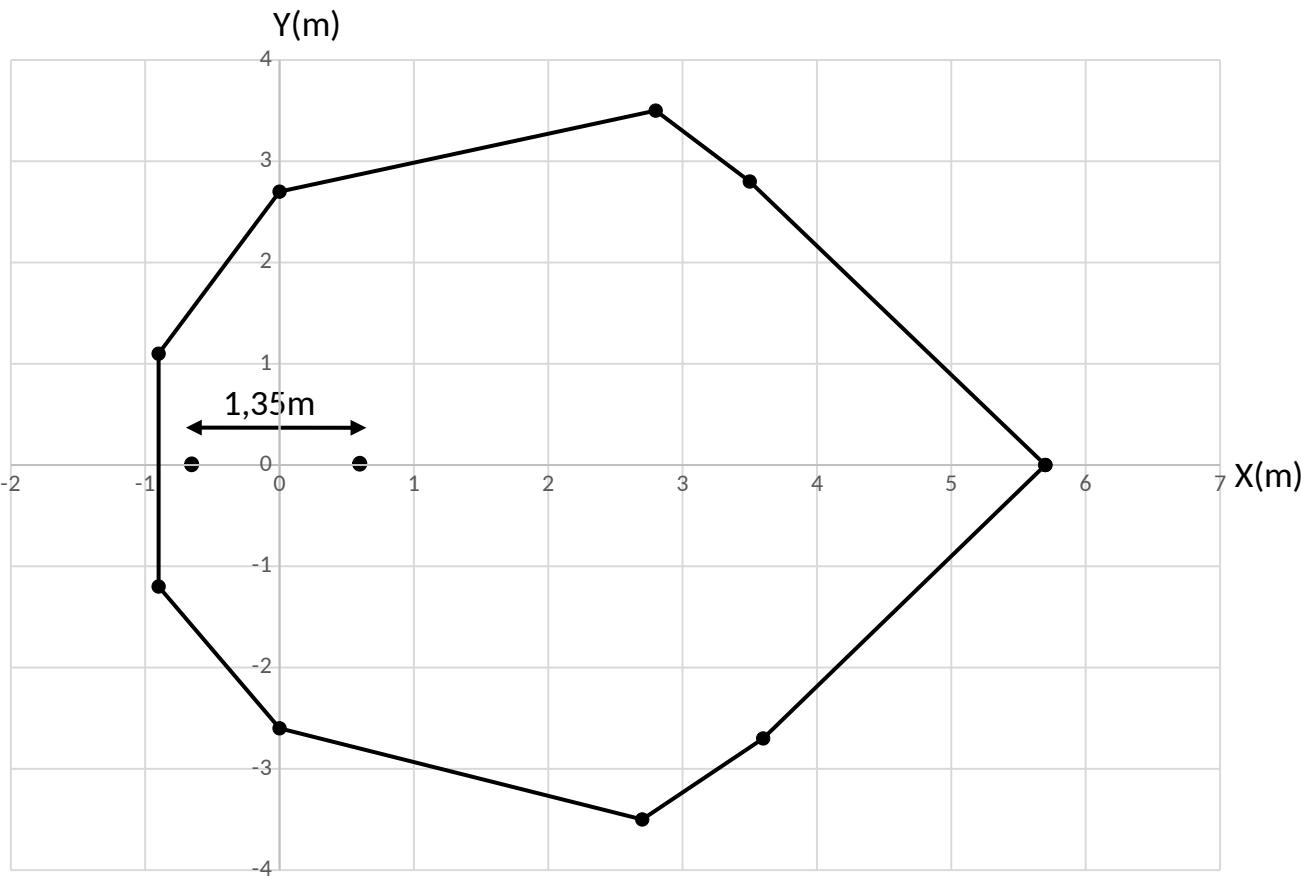
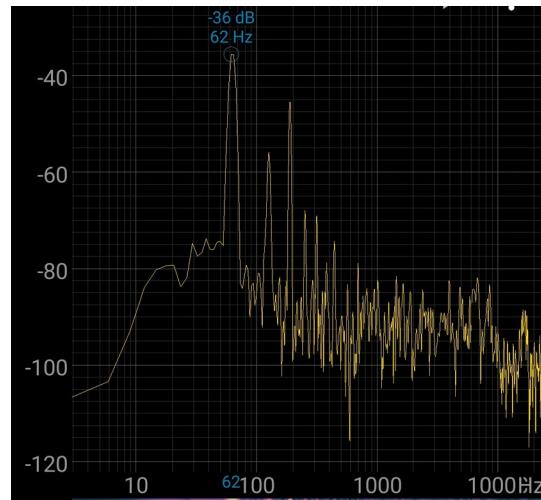


Valeurs obtenues (en m) :

x	y
-2,7	0
-2,4	2
0	2,8
2,1	2
3	0
2,1	-2
0	-3
-2,4	-2



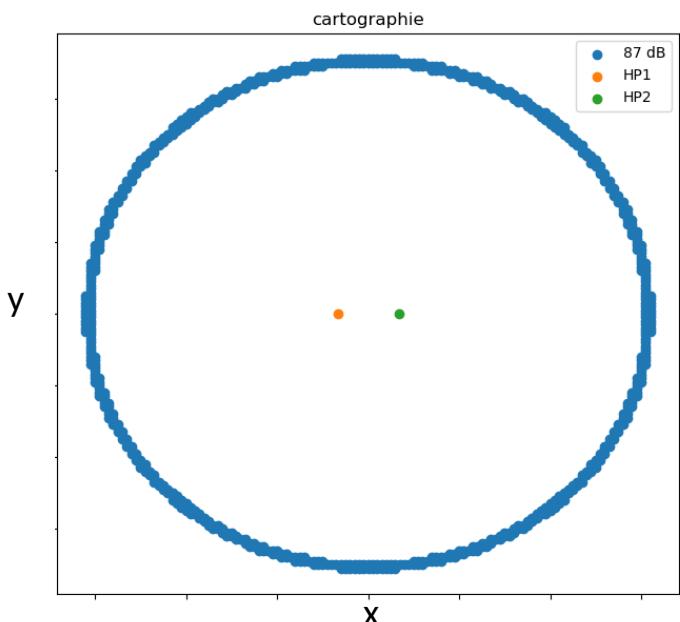
# Signal déphasé : courbe à niveau sonore constant



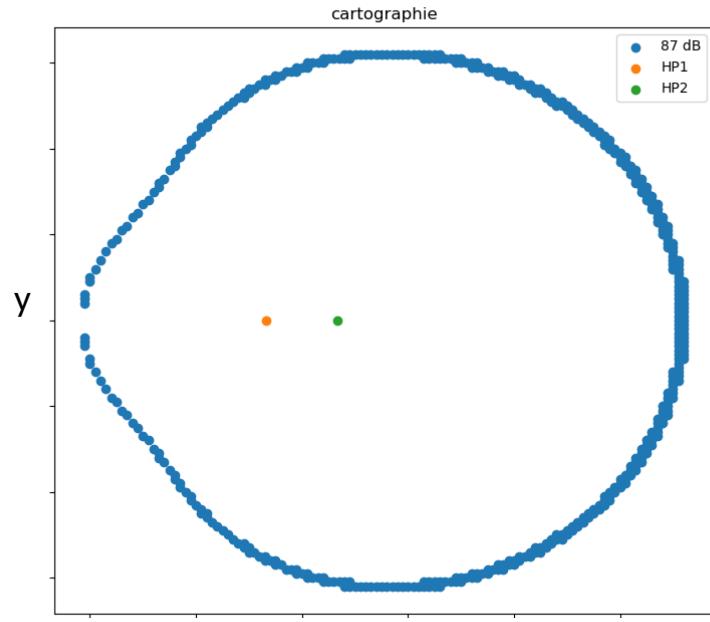
Valeurs obtenues (en m) :

x	y
5,7	0
3,5	2,8
2,8	3,3
0	2,7
-0,9	1,1
-0,9	-1,2
0	-2,6
2,7	-3,3
3,6	-2,7

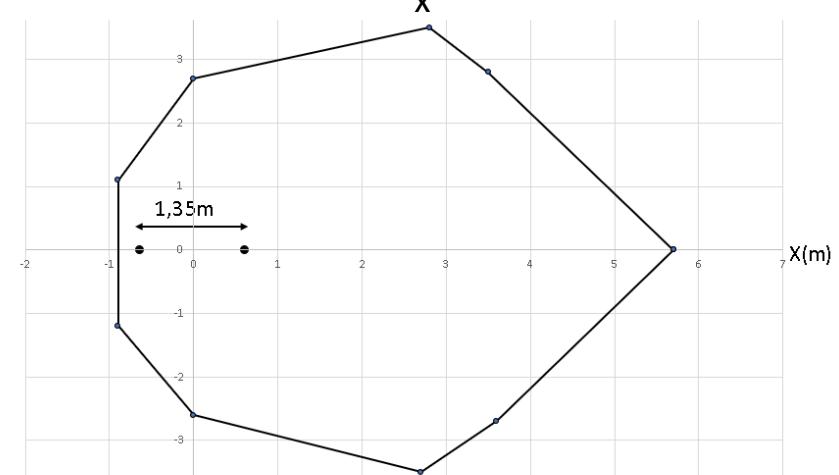
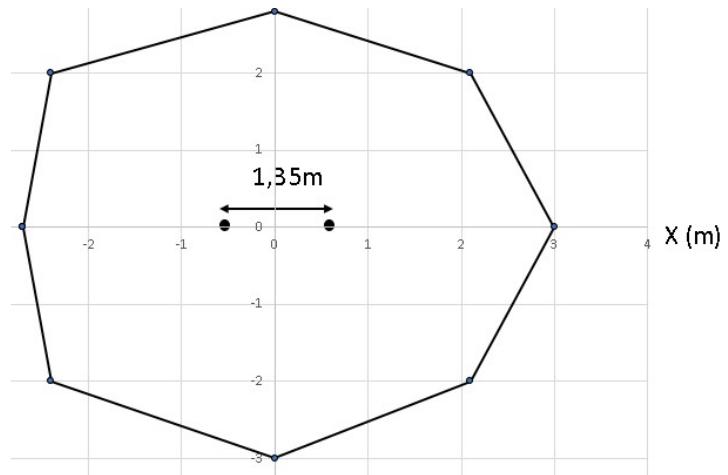
Signal non déphasé



Signal déphasé



Expérience



# Analyse - Conclusion

- Modélisation théorique cohérente
- Possibilité de contrôler partiellement la directivité
- Montage retenu :  $h = 1,35m$  et déphasage de  $\pi/2$
- Efficacité du montage étudié démontrée

# Bibliographie

Robert Harley : *The Complete Guide to High-End Audio*

<http://www.lafontaudio.com/divers/couplage.pdf>

<http://www.techniquesduson.com/sub.html>

<http://ilm-perso.univ-lyon1.fr/~pvincent/doaudio/cours.pdf>

<https://bv.univpoitiers.fr/access/content/user/fwatteau/site/peda/licence/M41/cm/enceinte.pdf>

## Logiciels spécifiques utilisés :

*Mapp XT (Meyer sound)*

*Spectroid*

*Audacity*

## Source des illustrations :

Schéma sur les interférences : Sciences-questions.org

Diagrammes de directivité : <http://ilm-perso.univ-lyon1.fr/~pvincent/docaudio/cours.pdf>

## Source de la formule de superposition :

[http://public.iut-en-ligne.net/electronique/piou\\_fruitet\\_fortun/baselecpro/acquisition/pdf/DL-001051-04-04.01.00.pdf](http://public.iut-en-ligne.net/electronique/piou_fruitet_fortun/baselecpro/acquisition/pdf/DL-001051-04-04.01.00.pdf)

# Détails des calculs

Notons  $L_A$  et  $L_B$  les distances respectives des points A et B avec le point P

$$L_A = \sqrt{\left(-\frac{h}{2} - x_P\right)^2 + (y_P)^2}$$

$$L_B = \sqrt{\left(\frac{h}{2} - x_P\right)^2 + (y_P)^2}$$

$$\Delta l = l_2 - l_1 = \sqrt{\left(-\frac{h}{2} - x_P\right)^2 + (y_P)^2} - \sqrt{\left(\frac{h}{2} - x_P\right)^2 + (y_P)^2}$$

$$\text{On en déduit le déphasage : } \varphi = \frac{\Delta l * f * 2\pi}{V_{son}} = \frac{\sqrt{\left(-\frac{h}{2} - x_P\right)^2 + (y_P)^2} - \sqrt{\left(\frac{h}{2} - x_P\right)^2 + (y_P)^2} * f * 2\pi}{V_{son}}$$

Niveau sonore au point P en isolant le HPa :

$$L(P)_A = L_{A1m} - 20 \log (I_{AP}) = L_{A1m} - 20 \log \left( \sqrt{\left(-\frac{h}{2} - x_P\right)^2 + (y_P)^2} \right)$$

$$L(P)_B = L_{B1m} - 20 \log (I_{BP}) = L_{B1m} - 20 \log \left( \sqrt{\left(\frac{h}{2} - x_P\right)^2 + (y_P)^2} \right)$$

On veut maintenant additionner ces niveaux sonores en tenant compte des interférences :  
 Dans un souci de linéarité, on utilise les intensités I sonores au lieu des niveaux sonores :

$$L = 10 * \log\left(\frac{I}{I_0}\right) \Leftrightarrow I = 10^{\frac{L}{10}} * I_0 \quad \text{Avec } I_0 = 10^{-12} \text{ W.m}^{-2}$$

Ainsi on a :

$$I_A(P) = 10^{\frac{L_A(P)}{10}} * I_0 = 10^{\frac{L_{A1m} - 20 \log(\sqrt{(\frac{h}{2} - x_P)^2 + (y_P)^2})}{10}} * I_0$$

$$I_B(P) = 10^{\frac{L_B(P)}{10}} * I_0 = 10^{\frac{L_{B1m} - 20 \log(\sqrt{(\frac{h}{2} - x_P)^2 + (y_P)^2})}{10}} * I_0$$

Superposition de signaux périodiques :

Soit  $u(t) = U \cos(\omega t)$  et  $v(t) = V \cos(\omega t + \varphi)$

$$u(t) + v(t) = \sqrt{A^2 + B^2} * \cos(\omega t)$$

Avec :  $A = U + V \cos(\varphi)$  et  $B = V \sin(\varphi)$

$$u(t) + v(t) = \sqrt{(U + V \cos(\varphi))^2 + (V \sin(\varphi))^2} * \cos(\omega t)$$

Notons  $I_R(P)$  l'intensité sonore résultante :  $I_R(P) = \sqrt{(I_A + I_B \cos(\varphi))^2 + (I_B \sin(\varphi))^2}$

J'en déduis le niveau sonore résultant au point P :

$$L_R(P) = 10 * \log\left(\frac{I_R(P)}{I_0}\right) = 10 * \log\left(\frac{\sqrt{(I_A + I_B \cos(\varphi))^2 + (I_B \sin(\varphi))^2}}{I_0}\right) = 10 * \log\left(\frac{\sqrt{(I_A + I_B \cos(\varphi))^2 + (I_B \sin(\varphi))^2}}{I_0}\right)$$

On obtient ainsi une expression du niveau sonore en tout point P(x<sub>p</sub>, y<sub>p</sub>) en fonction de f, h, L<sub>A1m</sub> et L<sub>B1m</sub> :

$$L_r(P) =$$

$$10 * \log\left(\frac{\frac{L_{A1m} - 20 \log\left(\sqrt{(x_p)^2 + \left(\frac{h}{2} - y_p\right)^2}\right)}{10} * I_0 + 10 + \frac{L_{B1m} - 20 \log\left(\sqrt{(x_p)^2 + \left(-\frac{h}{2} - y_p\right)^2}\right)}{10} * I_0 * \cos\left(\frac{\sqrt{(x_p)^2 + \left(\frac{h}{2} - y_p\right)^2} - \sqrt{(x_p)^2 + \left(\frac{h}{2} - y_p\right)^2} * f * 2\pi}{V_{son}}\right)}{I_0^2 + \left(\frac{L_{B1m} - 20 \log\left(\sqrt{(x_p)^2 + \left(-\frac{h}{2} - y_p\right)^2}\right)}{10} * I_0 * \sin\left(\frac{\sqrt{(x_p)^2 + \left(-\frac{h}{2} - y_p\right)^2} - \sqrt{(x_p)^2 + \left(\frac{h}{2} - y_p\right)^2} * f * 2\pi}{V_{son}}\right)\right)^2}\right)$$

Avec : I<sub>0</sub> = 10<sup>-12</sup> W.m<sup>-2</sup> et V<sub>son</sub> = 340m.s<sup>-1</sup>

On ajoute maintenant à HP<sub>B</sub> un délai Δt :

Cela revient à augmenter la différence de marche :

$$\Delta l = (l_2 - l_1) + V_{son} * \Delta t$$

On modifie donc l'expression de φ, puis de L<sub>R</sub>(P).

Ainsi on exprime L<sub>R</sub>(P) en fonction de P(x<sub>p</sub>, y<sub>p</sub>), f, h, L<sub>A1m</sub>, L<sub>B1m</sub> et Δt.

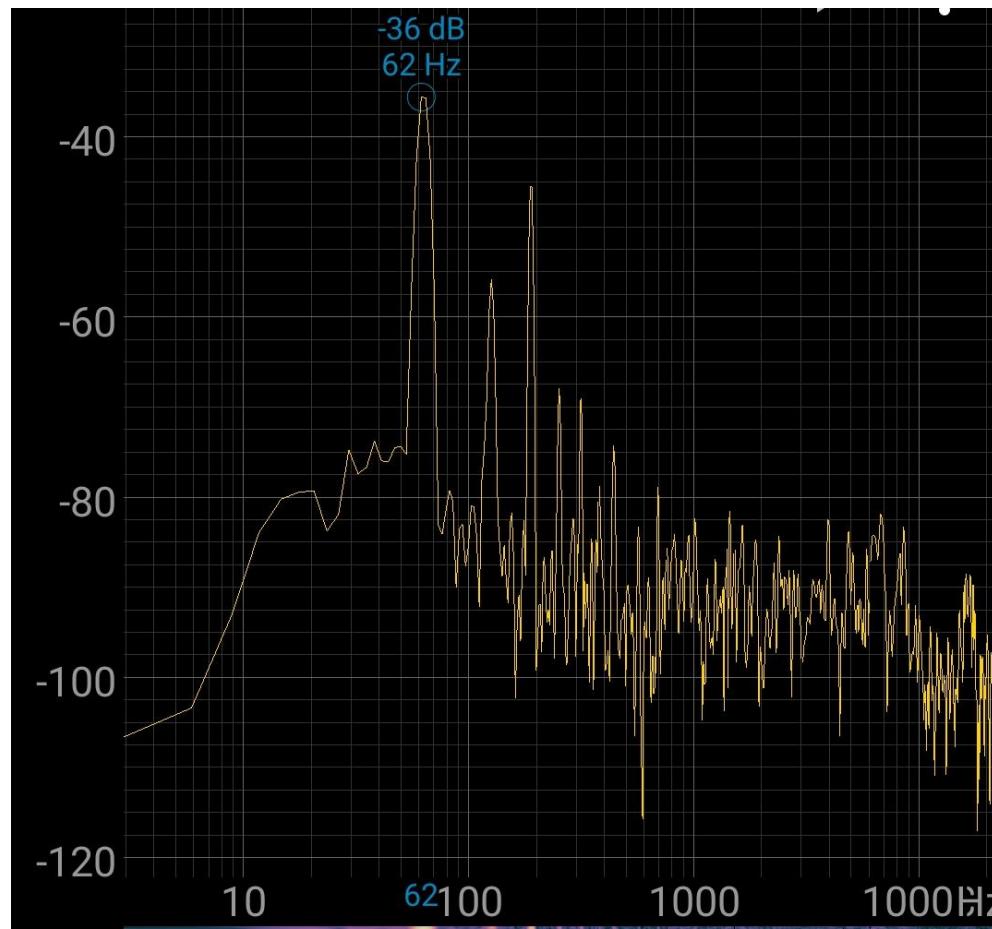
## Vérification de l'hypothèse : source sonore omnidirectionnelle

Expérience : Mesure du niveau sonore devant et derrière un seul haut-parleur

Résultats obtenus :

	Derrière le haut-parleur :	Devant le haut-parleur :
1 mètre	-35 dB	-31 dB
2 mètres	-37 dB	-36 dB
3 mètres	-39 dB	-38 dB
4 mètres	-40 dB	-40 dB
5 mètres	-41 dB	-41 dB

## Spectrogramme affiché lors des mesures



# Listing : Programme Python

```
# C:\Users\lucas\Desktop\TIPE\programme final.py
001| ##Bibliothèque
002|
003| import matplotlib.pyplot as plt
004| import numpy as np
005|
006| ##Fonctions
007| #fonctions servant au calcul du niveau sonore
008|
009| def niveau_sonore_1HP(X,Y,HP): #HP contient les caractéristiques d'un haut-
parleur : HP = [abscisse,ordonnée,L1m, délai]
010|     d = np.sqrt((X-HP[0])**2+(Y-HP[1])**2)
011|     L = HP[2]-20*np.log10(d)
012|     return L
013|
014| def difference_de_marche(X,Y,HP1,HP2):
015|     d1 = np.sqrt((HP1[1]-Y)**2+(HP1[0]-X)**2)
016|     d2 = np.sqrt((HP2[1]-Y)**2+(HP2[0]-X)**2)
017|     différence = d2-d1
018|     return différence
019|
020| def distance_delai(HP1,HP2):
021|     delta_t = HP2[3]-HP1[3]
022|     d = Vson*delta_t
023|
024|     return d
025|
026| def dephasage(f,X,Y,HP1,HP2):
027|     Vson = 340
028|     d = difference_de_marche(X,Y,HP1,HP2)+distance_delai(HP1,HP2)
029|     phi = (d*f**2*np.pi)/Vson
030|     phi = phi%(2*np.pi)
031|     return phi
032|
033| def niveau_sonore_2HP(X,Y,HP1,HP2): #niveau soore généralisé à 2HP
034|     L1 = niveau_sonore_1HP(X,Y,HP1)
035|     L2 = niveau_sonore_1HP(X,Y,HP2)
036|     phi = dephasage(f,X,Y,HP1,HP2)
037|     I1 = 10**(L1/10)*(10**(-12))
038|     I2 = 10**(L2/10)*(10**(-12))
039|     I_resultant = np.sqrt((I1+I2*np.cos(phi))**2+(I2*np.sin(phi))**2)
040|     L_resultant = 10*np.log10(I_resultant/(10**(-12)))
041|     return L_resultant
042|
066| #fonctions servant à la représentation graphique
067|
068| def cartographie(AmplitudeX,AmplitudeY, pas):
069|     Liste_coordonnées = []
070|     X0 = -AmplitudeX/2
071|     Y0 = -AmplitudeY/2
072|     nX = int(AmplitudeX/pas)
073|     nY = int(AmplitudeY/pas)
074|     x = X0
075|     for i in range(nX):
076|         y = Y0
077|         for j in range(nY):
078|             Point = []
079|             Point.append(x)
080|             Point.append(y)
081|             Liste_coordonnées.append(Point)
082|             Point.append(niveau_sonore_2HP(Point[0],Point[1],HP1,HP2)) #modif
083|             y = y + pas
084|             x = x + pas
085|     return Liste_coordonnées
086|
087|
088|     return Liste_L
089|
090| def selection(Liste_L, min, max): #modif
091|     Pts_selectionnés = []
092|     for i in range(len(Liste_L)):
093|         if min<Liste_L[i][2]<max:
094|             Pts_selectionnés.append(Liste_L[i])
095|     return Pts_selectionnés
096|
097|
098|
099| ##Programme principal
100| #Variables
101| f = 63
102| Vson = 340
103| HP1 = [0,-1.35/2,100,0]
104| HP2 = [0,1.35/2,100,0.004]#modif
105|
106| #Tracé
107| L = cartographie(17,28,0.1)
108| S = selection(L,83.9,84.1)
109|
110| abscisses = []
111| ordonnées = []
112| for i in range(len(S)):
113|     abscisses.append(S[i][0])
114|     ordonnées.append(S[i][1])
115|
116|
117| plt.scatter(abscisses,ordonnées, label = "87 dB")
118| plt.scatter(HP1[0],HP1[1], label = "HP1")
119| plt.scatter(HP2[0],HP2[1],label ="HP2")
120| plt.legend()
121| plt.title("cartographie")
122| axis = ("equal")
123| plt.show()
```